

The Computational Complexity of Rules for the Character Table of S_n

Dan Bernstein

Department of Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel

Abstract

The Murnaghan-Nakayama rule is the classical formula for computing the character table of S_n . Y. Roichman (Roichman 1997) has recently discovered a rule for the Kazhdan-Lusztig characters of q Hecke algebras of type A , which can also be used for the character table of S_n . For each of the two rules, we give an algorithm for computing entries in the character table of S_n . We then analyze the computational complexity of the two algorithms, and in the case of characters indexed by partitions in the (k, ℓ) hook, compare their complexities to each other. It turns out that the algorithm based on the Murnaghan-Nakayama rule requires far less operations than the other algorithm. We note the algorithms' complexities' relation to two enumeration problems of Young diagrams and Young tableaux.

1 Introduction

This paper examines two formulas for computing entries in the character table (hereafter called *character values*) of the symmetric group, S_n , from the standpoint of computational complexity. The formulas that we consider are the classical Murnaghan-Nakayama rule (Murnaghan, 1937; Nakayama, 1940) and the rule recently discovered by Roichman (Roichman, 1997) for the Kazhdan-Lusztig characters of q Hecke algebras of type A . The discussion is motivated by a remark in Barcelo, Ram (1999), in which the authors state that they are unaware of a comparison of the two rules in terms of algorithmic complexity, and that “one would expect that they have the same complexity”.

The irreducible characters of S_n are a distinguished set of class functions $\{\chi^\lambda : S_n \rightarrow \mathbb{Z} \mid \lambda \vdash n\}$ (for a complete description see Sagan, 1991). A

Email address: `dan.bernstein@weizmann.ac.il` (Dan Bernstein).

character value of S_n is indexed by an ordered pair (λ, μ) of partitions of n and is denoted by

$$\chi^\lambda(\mu) = \chi^\lambda(w) \quad w \in S_n \text{ is of cycle type } \mu.$$

A formula for $\chi^\lambda(\mu)$ suggests a systematic way for computing character values — an algorithm whose input is a pair of partitions (λ, μ) and whose output is the integer $\chi^\lambda(\mu)$. It is such an algorithm's computational complexity that is examined for each of the two rules.

The rest of the paper is organized as follows: in section 2 we present the Murnaghan-Nakayama rule and specify an algorithm based on it. In section 3 we treat Roichman's rule similarly.

Section 4 gives the complexity of computing a single character value using each of the two algorithms. Two enumeration problems, of Young diagrams and of Young tableaux, occur in the dominant factors in the complexity of the Murnaghan-Nakayama rule (eq. 8) and Roichman's rule (eq. 9), respectively.

Finally, in section 5, we compare the algorithms in terms of their worst-case complexity on the family of characters indexed by partitions in the (k, ℓ) hook.

1.1 Main Results

Given n and partitions λ and μ of n we show that:

- (1) The running time of our Murnaghan-Nakayama-based algorithm is, up to a factor of order n , the number of Young diagrams that are contained in the Young diagram of λ and satisfy an additional constraint determined by μ (see proposition 8). Lemma 9 gives a determinantal formula for this number when the constraint is empty.
- (2) The running time of the algorithm based on Roichman's rule is, up to a factor of order n , the number of standard Young tableaux whose shape is contained in the Young diagram of λ that satisfy an additional constraint determined by μ (see proposition 10). By lemma 11, when the constraint is empty, this number is $O(n)d_\lambda$, where d_λ is the degree of λ , that is the number of standard Young tableaux of shape λ .

Given k and ℓ , worst-case analysis of the family of characters where the choice of λ is restricted to the (k, ℓ) hook shows that in this case the Murnaghan-Nakayama-based algorithm's complexity is $\Theta(n^{k+\ell+1})$ (see proposition 12) whereas the complexity of the algorithm based on Roichman's rule is in $\Omega(n^{-g}(k+\ell)^n) \cap O(n^{-g+2}(k+\ell)^n)$ for some constant g (see proposition 15).

Some experimental results for characters not in the above family are also included in subsection 5.2.



2 The Murnaghan-Nakayama Rule


A shape $A \subset \mathbb{N} \times \mathbb{N}$ is said to be *edgewise connected* if

$$A = \{(i_1, j_1), (i_2, j_2), \dots, (i_n, j_n)\}$$

and for all $k < n$

$$|i_k - i_{k+1}| + |j_k - j_{k+1}| = 1$$

(i.e. each cell is exactly one horizontal or vertical step away from its predecessor). For example,  is edgewise connected, but  is not.

Let $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m) \vdash n$. A skew diagram $\xi = \lambda/\mu$ is said to be a *rim hook* of λ if ξ is edgewise connected and contains no 2×2 subset of cells (). In this case we write $\lambda \setminus \xi = \mu$ and say that μ is obtained by *removing* the rim hook ξ from λ . For example, if $\lambda = (4, 3, 2)$, then

$$\lambda/(2, 2, 2) = \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \\ \hline \end{array}$$

is a rim hook of λ , but

$$\lambda/(2, 2, 1) = \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \\ \hline \square & \\ \hline \end{array} \quad \text{and} \quad \lambda/(1, 1) = \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \square \\ \hline \square & \square & \\ \hline \end{array}$$

are not: the former is not edgewise connected, and the latter contains a 2×2 block.

The *leg length* of a rim hook ξ is

$$ll(\xi) = (\text{the number of rows of } \xi) - 1.$$

Let $\lambda \setminus \lambda_1$ denote the partition $(\lambda_2, \lambda_3, \dots, \lambda_m)$.

Note that the notation λ/μ is reserved for skew diagrams, while $\lambda \setminus \xi$ and $\lambda \setminus \lambda_1$ are always ordinary diagrams.

The following is the classical recursive formula for computing characters of S_n .

Theorem 1 (The Murnaghan-Nakayama Rule) *Let $\lambda, \mu \vdash n$. Then*

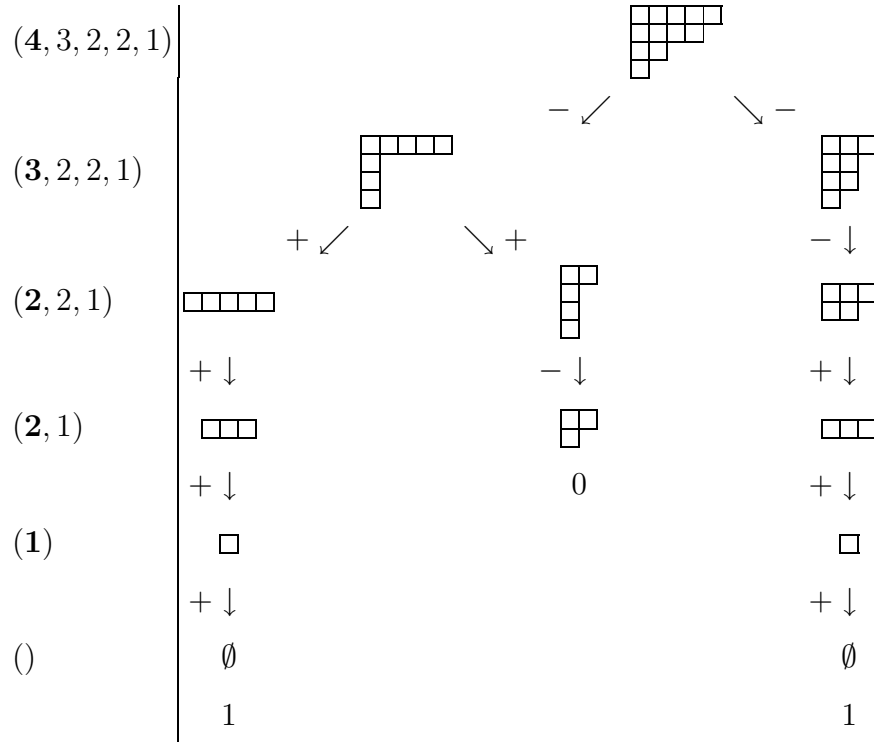
$$\chi^\lambda(\mu) = \sum_{\xi} (-1)^{l(\xi)} \chi^{\lambda \setminus \xi}(\mu \setminus \mu_1) \quad (1)$$

where the sums runs over all rim hooks ξ of λ having μ_1 cells, and $\chi^{(0)}(0) = 1$.

A proof appears in Sagan (1991).

Example 2 *Calculating $\chi^\lambda(\mu)$ where $\lambda = (5, 4, 2, 1)$ and $\mu = (4, 3, 2, 2, 1)$.*

The computation process can be viewed as a tree. The appropriate signs appear beside the arrows indicating the removal of rim hooks.



$$\begin{aligned}
\chi^{(5,4,2,1)}(4,3,2,2,1) &= -\chi^{(5,1,1,1)}(3,2,2,1) - \chi^{(3,2,2,1)}(3,2,2,1) \\
&= -(\chi^{(5)}(2,2,1) + \chi^{(2,1,1,1)}(2,2,1)) + \chi^{(3,2)}(2,2,1) \\
&= -(\chi^{(3)}(2,1) - \chi^{(2,1)}(2,1)) + \chi^{(3)}(2,1) \\
&= -(\chi^{(1)}(1) - 0) + \chi^{(1)}(1) \\
&= -(1 + 0) + 1 \\
&= 0.
\end{aligned}$$

2.1 An algorithm based on the Murnaghan-Nakayama rule

Computing the sum in (1) requires enumerating all rim hooks of certain length of a given partition. This is done using partition sequences (Olsson (1993), Bessenrodt (1998)).

A *partition sequence* Λ is a doubly infinite sequence of binary digits starting with an infinite sequence of zeros and ending with an infinite sequence of ones. For example,

$$\Lambda = \dots \mathbf{00} \underbrace{\mathbf{101011010}}_{\bar{\Lambda}} \mathbf{11} \dots$$

where the dots at the beginning (end) represent an infinite sequence of **0**s (**1**s), is a partition sequence.

We shall refer to the finite subsequence of Λ starting with the first $\mathbf{1}$ and ending with the last $\mathbf{0}$ as the *essential part of* Λ , which we will denote by $\bar{\Lambda}$.

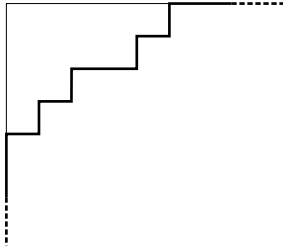
Given a partition $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)$, its partition sequence is defined as

$$\Lambda = \dots 00 \underbrace{11 \dots 1}_{{\lambda_m}} 0 \underbrace{11 \dots 1}_{{\lambda_{m-1}-\lambda_m}} 0 \underbrace{11 \dots 1}_{{\lambda_{m-2}-\lambda_{m-1}}} 0 1 \dots 1 0 \underbrace{11 \dots 1}_{{\lambda_2-\lambda_1}} 0 1 1 \dots$$

For example, the partition sequence of $\lambda = (5, 4, 2, 1)$ is

$$\Lambda = \dots 00\underline{1}0\underline{1}0\underline{1}10\underline{1}0\underline{1}1\dots$$

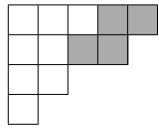
The graphic version of this construction is a walk along the borderline of λ , coming from the south on the vertical line, going along the border and leaving on the horizontal line eastwards, recording each vertical step by a **0** and each horizontal step by a **1**. In our example, the borderline of the Young diagram of $\lambda = (5, 4, 2, 1)$ is



which indeed gives the sequence

$$\dots 0010101101011\dots = \Lambda.$$

Consider the rim hook ξ of $\lambda = (5, 4, 2, 1)$:



The partition sequence of $\mu = \lambda \setminus \xi = (3, 2, 2, 1)$ is

$$M = \dots \mathbf{0} \mathbf{0} \mathbf{1} \mathbf{0} \mathbf{1} \mathbf{0} \boxed{\mathbf{0}} \mathbf{1} \mathbf{0} \mathbf{1} \boxed{\mathbf{1}} \mathbf{1} \mathbf{1} \dots$$

and the partition sequence of λ is

$$\Lambda = \dots \mathbf{0} \mathbf{0} \mathbf{1} \mathbf{0} \mathbf{1} \mathbf{0} \boxed{\mathbf{1}} \mathbf{1} \mathbf{0} \mathbf{1} \boxed{\mathbf{0}} \mathbf{1} \mathbf{1} \dots$$

We observe that M differs from Λ only by the exchange with one another of the two digits in the positions marked in the above sequences, changing their order from the $\mathbf{1}$ being to the left of the $\mathbf{0}$ in Λ , to the $\mathbf{0}$ being to the left of the $\mathbf{1}$ in M . Moreover, we note that the two digits are $4 = |\xi|$ positions apart from each other and that there is exactly $1 = \ell(\xi)$ $\mathbf{0}$ between them.

This is not by coincidence, as the following definitions and the next proposition show.

Let Λ be a partition sequence. A *rim hook in Λ* is a pair consisting of a $\mathbf{0}$ and a $\mathbf{1}$ in Λ such that the $\mathbf{1}$ appears to the left of the $\mathbf{0}$. The distance between the $\mathbf{0}$ and the $\mathbf{1}$ is the *length* of the rim hook, and the number of $\mathbf{0}$ s strictly between them is the *leg length* of the rim hook. The rim hook is *removed* by exchanging the $\mathbf{0}$ with the $\mathbf{1}$. For example, the marked pair of digits in

$$\Lambda = \dots \mathbf{0} \mathbf{0} \mathbf{1} \mathbf{0} \boxed{\mathbf{1}} \mathbf{0} \mathbf{1} \mathbf{1} \mathbf{0} \mathbf{1} \boxed{\mathbf{0}} \mathbf{1} \mathbf{1} \dots$$

is a rim hook of length 6 and of leg length 2. The partition sequence obtained from Λ by removing this rim hook is

$$\dots \mathbf{0} \mathbf{0} \mathbf{1} \mathbf{0} \boxed{\mathbf{0}} \mathbf{0} \mathbf{1} \mathbf{1} \mathbf{0} \mathbf{1} \boxed{\mathbf{1}} \mathbf{1} \mathbf{1} \dots$$

Proposition 3 *There is a bijection between rim hooks in the partition sequence Λ of a partition λ and rim hooks of the Young diagram of λ . Moreover, this bijection preserves the notions of length and leg length, and the removal of a rim hook in the partition sequence corresponds to the removal of the corresponding rim hook of the Young diagram.*

Based on this, the following algorithm, *MNinner*, finds all rim hooks ξ of λ having length μ_1 simply by going over all pairs of digits that are μ_1 places apart from each other in $\bar{\Lambda}$, where Λ is the partition sequence of λ . A variable

σ keeps track of $(-1)^{\# \text{ of } 0\text{s between the 2 digits of the pair}}$. If and only if the left digit in such a pair is **1** and the right digit is **0**, then it is a rim hook, and then the partition sequence of $\lambda \setminus \xi$ is obtained by exchanging the **0** with the **1**. *MNinner* then proceeds recursively to compute $\chi^{\lambda \setminus \xi}(\mu \setminus \mu_1)$ and adds $\sigma \chi^{\lambda \setminus \xi}(\mu \setminus \mu_1)$ to the sum.

Function *MNinner* (R, t)

Input: A sequence of binary digits $R = R_1 R_2 \dots R_s$ and an index t

Output: $\chi^\rho(\nu)$ where ρ is the partition whose partition sequence

is $\dots \mathbf{0} R \mathbf{1} \dots$ and $\nu = (\mu_t, \mu_{t+1}, \dots, \mu_k)$.

```

if  $t > k$ 
  then  $\chi \leftarrow 1$ 
else  $\chi \leftarrow 0$ 
   $\sigma \leftarrow 1$ 
  for  $j \leftarrow 1$  to  $\mu_t - 1$ 
    do if  $R_j = 0$  then  $\sigma \leftarrow -\sigma$ 
  for  $i \leftarrow 1$  to  $s - \mu_t$ 
    do if  $R_i \neq R_{i+\mu_t-1}$  then  $\sigma \leftarrow -\sigma$ 
      if  $(R_i, R_{i+\mu_t})$  is a rim hook,  $\xi$ 
        then exchange between  $R_i$  and  $R_{i+\mu_t}$ .
           $\chi \leftarrow \chi + \sigma \cdot \text{MNinner}(R, t + 1)$ 
          exchange between  $R_i$  and  $R_{i+\mu_t}$ .
  return  $\chi$ 

```

A major inefficiency of *MNinner* is that character values that occur more than once in the expansion of the right hand side of (1) are re-computed each time. In example 2, $\chi^{(3)}(2, 1)$ occurs twice (and therefore so does $\chi^{(1)}(1)$), so *MNinner* is invoked twice to compute it. This is overcome in the following algorithm, *MNinner*, by saving intermediate results in a table and using it to look up character values before computing them. Each time a value $\chi^\rho(\nu)$ is computed, it is recorded in a table T , and each time a value is required, it is first looked up in T , and only if it is not there, then it is computed. T is indexed by partitions ρ rather than by pairs (ρ, ν) of partitions, since for any $\chi^\rho(\nu)$ appearing in the expansion of (1) we have that ν is the tail of μ of weight $|\rho|$.

Function $MN1inner(R, t)$

Input: A sequence of binary digits $R = R_1 R_2 \dots R_s$ and an index t

Output: $\chi^\rho(\nu)$ where ρ is the partition whose partition sequence is $\dots \mathbf{0} R \mathbf{1} \dots$ and $\nu = (\mu_t, \mu_{t+1}, \dots, \mu_k)$.

```

if  $t > k$ 
  then  $\chi \leftarrow 1$ 
else  $\chi \leftarrow 0$ 
   $\sigma \leftarrow 1$ 
  for  $j \leftarrow 1$  to  $\mu_t - 1$ 
    do if  $R_j = 0$  then  $\sigma \leftarrow -\sigma$ 
  for  $i \leftarrow 1$  to  $s - \mu_t$ 
    do if  $R_i \neq R_{i+\mu_t-1}$  then  $\sigma \leftarrow -\sigma$ 
      if  $(R_i, R_{i+\mu_t})$  is a rim hook,  $\xi$ 
        then exchange between  $R_i$  and  $R_{i+\mu_t}$ .
          if  $T(\tilde{\rho})$  is empty, where  $\tilde{\rho} = \rho \setminus \xi$  is the partition
            whose partition sequence is  $\dots \mathbf{0} R \mathbf{1} \dots$ 
              then  $T(\tilde{\rho}) \leftarrow MN1inner(R, t + 1)$ 
           $\chi \leftarrow \chi + \sigma T(\tilde{\rho})$ 
          exchange between  $R_i$  and  $R_{i+\mu_t}$ .
return  $\chi$ 

```

Given partitions λ and μ , to compute $\chi^\lambda(\mu)$ one needs to compute the essential part of λ 's partition sequence, $\bar{\Lambda}$, and then to invoke $MN1inner(\bar{\Lambda}, 1)$. This is what algorithms *PartSeq* and *MurNak* do.

Function $PartSeq(\lambda)$

Input: a partition $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)$. $m = \ell(\lambda)$

Output: $\bar{\Lambda}$, the essential part of the partition sequence of λ

```

 $\bar{\Lambda} \leftarrow$  an empty sequence
 $\lambda_{m+1} \leftarrow 0$ 
for  $i \leftarrow m$  down to 1
  do for  $k \leftarrow 1$  to  $\lambda_i - \lambda_{i+1}$ 
    do  $\bar{\Lambda} \leftarrow \bar{\Lambda} \parallel 1$ 
   $\bar{\Lambda} \leftarrow \bar{\Lambda} \parallel 0$ 
return  $\bar{\Lambda}$ 

```

Function $MurNak(\lambda, \mu)$

Input: partitions λ and μ of the same weight

Output: $\chi^\lambda(\mu)$

```

 $T \leftarrow$  a 1-dimensional sparse array
 $\bar{\Lambda} \leftarrow PartSeq(\lambda)$ 
 $\chi \leftarrow MN1inner(\bar{\Lambda}, 1)$ 
return  $\chi$ 

```


3 Roichman's Rule

Let (W, S) be a Coxeter system, and let $\ell(w)$, $w \in W$ be the length function with respect to S .

The q Hecke algebra \mathcal{H} of W is the algebra spanned by the set $\{T_w \mid w \in W\}$ over the ring of Laurent polynomials $\mathbb{Z}[q, q^{-1}]$ subject only to the following relations:

$$\begin{aligned} T_s T_w &= T_{sw} \quad \text{if } s \in S \text{ and } \ell(sw) > \ell(w) \\ T_s^2 &= (q - 1)T_s + qT_1 \quad \text{if } s \in S \end{aligned}$$

where T_1 acts as the identity.

Kazhdan, Lusztig (1979) gives a distinguished basis $\{C_w \mid w \in W\}$ for \mathcal{H} and a partition of the Coxeter group W into *Kazhdan-Lusztig cells*. Each left Kazhdan-Lusztig cell \mathcal{C} has a left representation of \mathcal{H} associated to it. Let $\chi^{\mathcal{C}}$ be the character of that representation. Then for any $T \in \mathcal{H}$ and finite cell \mathcal{C}

$$\chi^{\mathcal{C}}(T) = \sum_{w \in \mathcal{C}} TC_w(w) \quad (2)$$

where $TC_w(w)$ is the coefficient of C_w in TC_w .

Roichman (1997) gives a formula for $T_{s_1 s_2 \dots s_k} C_w(w)$ where $s_1, \dots, s_k \in S$, subject to certain relations between the s_i . In the case $W = S_n$, the formula applies to all of the summands in (2). Furthermore in the $W = S_n$ case, the Kazhdan-Lusztig characters are exactly the irreducible characters, and the Robinson-Schensted-Knuth correspondence gives rise to a canonical map between the Kazhdan-Lusztig and Young's natural characters of S_n , allowing for the formulation of the character as a weighted sum over standard tableaux.

If $\mu = (\mu_1, \dots, \mu_k) \vdash n$, define $B(\mu) = \{\mu_1 + \dots + \mu_r \mid 1 \leq r \leq k\}$. For example, if $\mu = (5, 2, 1, 1)$ then $B(\mu) = \{5, 7, 8, 9\}$.

Recall that a *standard tableau* is a tableau whose rows and columns are increasing. The *descent set* of a standard tableau T is

$$D(T) = D(w(T)^{-1})$$

the descent set of the inverse of the reading word of T , also characterized by

$$D(T) = \{i \mid i + 1 \text{ is in the southwest of } i \text{ in } T\}$$

where “southwest” means strictly below and weakly to the left. For example, the descent set of $T = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 4 & 5 & 8 \\ \hline 3 & 6 & & & \\ \hline 7 & 9 & & & \\ \hline \end{array}$ is $\{2, 5, 6, 8\}$.

Define $f_\mu^q(T, i)$, $i = 1, 2, \dots, n$ by

$$f_\mu^q(T, i) = \begin{cases} -1 & i \notin B(\mu), i \in D(T) \\ 0 & i, i+1 \notin B(\mu), i \notin D(T) \text{ and } i+1 \in D(T) \\ q & \text{otherwise.} \end{cases} \quad (3)$$

Theorem 4 (Roichman's Rule (Roichman, 1997)) *Let $\lambda, \mu \vdash n$, and let χ^λ be the corresponding character of the q Hecke algebra of S_n . Let T_μ be any element in the Hecke algebra indexed by a permutation $w \in S_n$ of cycle type μ . Then*

$$\chi^\lambda(T_\mu) = \sum_T \prod_{\substack{1 \leq i < n \\ i \notin B(\mu)}} f_\mu^q(T, i)$$

where the sum runs over all standard tableaux T of shape λ .

Substituting 1 for q in the above we get a rule for the characters of S_n :

$$\chi^\lambda(\mu) = \sum_T \prod_{1 \leq i < n} f_\mu^1(T, i) \quad (4)$$

where the sum runs over all standard tableaux T of shape λ .

Example 5 *Calculating $\chi^\lambda(\mu)$ where $\lambda = (2, 1, 1)$ and $\mu = (3, 1)$. We have $B(\mu) = \{3, 4\}$ and*

T	$f_\mu^1(T, 1)$	$f_\mu^1(T, 2)$	$f_\mu^1(T, 3)$	$\prod_{1 \leq i < 4} f_\mu^1(T, i)$
$\begin{array}{ c c } \hline 1 & 2 \\ \hline 3 & \\ \hline 4 & \\ \hline \end{array}$	0	-1	1	0
$\begin{array}{ c c } \hline 1 & 3 \\ \hline 2 & \\ \hline 4 & \\ \hline \end{array}$	-1	1	1	-1
$\begin{array}{ c c } \hline 1 & 4 \\ \hline 2 & \\ \hline 3 & \\ \hline \end{array}$	-1	-1	1	1

Hence $\chi^\lambda(\mu) = 0 + (-1) + 1 = 0$.

3.1 Recursive formulation

The naïve way to compute $\chi^\lambda(\mu)$ using Roichman's rule would be to construct all standard tableaux of shape λ , and for each tableau T to compute the values of $f_\mu^1(T, i)$ for all i (or until a 0 value is encountered), and finally to take the products and their sum. However, it can easily be shown that $f_\mu^q(T, i)$ depends only on the first $i+2$ entries of T . An improvement over the naïve approach is achieved by using this observation, as follows:

Let T be a standard tableau. Denote by $T|_i$ the standard tableau obtained by deleting all entries $j > i$ from T . For example, $\begin{array}{|c|c|c|c|} \hline 1 & 2 & 4 & 6 \\ \hline 3 & 5 & 7 & \\ \hline \end{array}|_4 = \begin{array}{|c|c|c|} \hline 1 & 2 & 4 \\ \hline 3 & & \\ \hline \end{array}$.

Define

$$g_\mu(T, i) = \begin{cases} 1 & i = 1, 2 \\ f_\mu^1(T, i - 2) & i > 2. \end{cases}$$

Let Q be a standard tableau of shape $\alpha \subseteq \lambda$, $|\alpha| = j$. Define

$$A(\lambda, \mu, Q) = \sum_T \prod_{j < i < n+2} g_\mu(T|_i, i) \quad (5)$$

where the sum runs over all standard tableaux T of shape λ containing Q (i.e. such that $T|_j = Q$). Note that

$$\begin{aligned} A(\lambda, \mu, \emptyset) &= \sum_T \prod_{0 < i < n+2} g_\mu(T|_i, i) \\ &= \sum_T \prod_{0 < i < n+2} g_\mu(T|_i, i) \\ &= \sum_T \prod_{2 < i < n+2} f_\mu^1(T|_i, i - 2) \\ &= \sum_T \prod_{0 \leq i < n} f_\mu^1(T|_{i+2}, i) \\ &= \sum_T \prod_{0 \leq i < n} f_\mu^1(T, i) \end{aligned}$$

where the sums run over all standard tableaux of shape T , so (4) can be rewritten as:

$$\chi^\lambda(\mu) = A(\lambda, \mu, \emptyset). \quad (6)$$

The following proposition follows easily from the definitions.

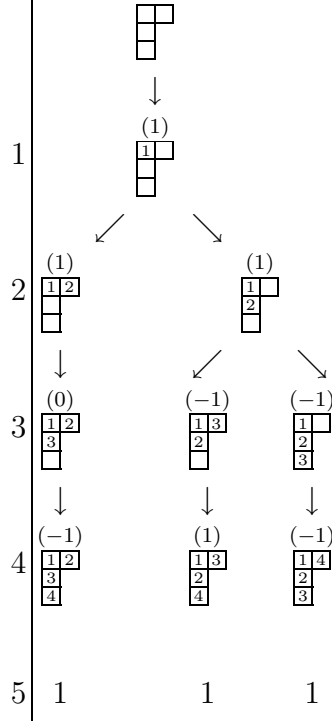
Proposition 6 *We have the following recursive formula:*

$$A(\lambda, \mu, Q) = \begin{cases} g_\mu(Q, n+1) & j = n \\ \sum_S g_\mu(S, j+1) A(\lambda, \mu, S) & \text{otherwise} \end{cases} \quad (7)$$

where the sum runs over all standard tableaux S such that $\text{sh}(S) \subseteq \lambda$, $Q \subset S$ and $|S| = |Q| + 1$, i.e. the tableaux S are those obtained by adding $j+1$ to Q in a position belonging to λ .

Example 7 *Calculating $\chi^\lambda(\mu)$ where $\lambda = (2, 1, 1)$ and $\mu = (3, 1)$, hence $B(\mu) = \{3, 4\}$. Each node in the following tree shows a tableau Q of some shape contained in λ , starting with the empty tableau. Each node's children are all the tableaux S of shape contained in λ that can be obtained by adding one entry to Q . The number in parentheses above each tableau Q is $g_\mu(Q, j)$*

where j is the number of entries in Q , also appearing in the column to the left of the tree. The numbers in the last row are $g_\mu(\cdot, 5)$ for the tableaux above them.



$$\chi^{(2,1,1)}(3, 1) = 1(1 \cdot 0 + 1(-1 \cdot 1 \cdot 1 + (-1)(-1)1)) = 0.$$

3.2 An algorithm based on Roichman's rule

The following algorithm, *RoiInner*, computes $A(\lambda, \mu, Q)$ according to proposition 6, computing values of g_μ and invoking itself recursively as necessary. It assumes that the global variable $B = B_1 B_2 \dots B_n$ is assigned the values $B_m = 1_{B(\mu)}(m)$. In the case $j = n$ of (7), it computes $g_\mu(Q, n+1) = f_\mu^1(Q, n-1)_{|q=1}$ by checking for the first case of (3) (note that the second case of (3) cannot occur for $i = n-1$ since $n \in B(\mu)$ always). In the case $j < n$ of (7), the algorithm computes the sum by going over the rows of Q , checking for each row whether by adding $j+1$ at its end one gets a tableau S such that $\text{sh}(S) \subset \lambda$. If so, it sets the variable \tilde{d} to indicate whether $j \in D(S)$ and determines $g_\mu(S, j+1)$, which is assigned to the variable g . Finally, if $g \neq 0$, it proceeds recursively to compute $A(\lambda, \mu, S)$ and adds $g_\mu(S, j+1)A(\lambda, \mu, S)$ to the sum.

Function $RoiInner(\alpha, j, m, d)$

Input: a partition $\alpha = (\alpha_1, \dots, \alpha_\ell)$, $j = |\alpha|$, a row index m and an indicator d .

Output: $A(\lambda, \mu, Q)$ where Q is any of the tableaux such that

$sh(Q) = \alpha$, j appears on row m of Q and d indicates whether $j - 1 \in D(Q)$.

```

if  $j = n$ 
  then if ( $d = \text{yes}$  and  $B_{n-1} = 0$ )
    then  $A \leftarrow -1$ 
    else  $A \leftarrow 1$ 
  else  $A \leftarrow 0$ 
  for  $k \leftarrow 1$  to  $\ell$ 
    do if ( $(k = 1$  or  $\alpha_k < \alpha_{k-1})$  and  $\alpha_k < \lambda_k$ )
      then if  $k > m$ 
        then  $\tilde{d} \leftarrow \text{yes}$ 
        else  $\tilde{d} \leftarrow \text{no}$ 
      if ( $j + 1 < 3$  or  $B_{j-1} = 1$ )
        then  $g \leftarrow 1$ 
        else if  $d = \text{yes}$ 
          then  $g \leftarrow -1$ 
          else if ( $\tilde{d} = \text{yes}$  and  $d = \text{no}$  and  $B_j = 0$ )
            then  $g \leftarrow 0$ 
            else  $g \leftarrow 1$ 
      if  $g \neq 0$ 
        then  $\alpha_k \leftarrow \alpha_k + 1$ 
         $A \leftarrow A + g \cdot RoiInner(\alpha, j + 1, k, \tilde{d})$ 
         $\alpha_k \leftarrow \alpha_k - 1$ 

return  $A$ 

```

Given partitions λ and μ , to compute $\chi^\lambda(\mu)$ one needs to initialize the global variable B to contain $B(\mu)$ and then to compute $A(\lambda, \mu, \emptyset)$ using $RoiInner$. This is what the algorithm *Roich* does.

Function $Roich(\lambda, \mu)$

Input: partitions $\lambda = (\lambda_1, \dots, \lambda_\ell)$ and $\mu = (\mu_1, \dots, \mu_k)$ of the same weight. $\ell = \ell(\lambda)$

Output: $\chi^\lambda(\mu)$

$\alpha \leftarrow \underbrace{(0, 0, \dots, 0)}_\ell$

```

for  $i \leftarrow 1$  to  $k$ 
  do for  $j \leftarrow 1$  to  $\mu_i - 1$ 
    do  $B \leftarrow B \parallel 0$ 
     $B \leftarrow B \parallel 1$ 

```

$\chi \leftarrow RoiInner(\alpha, 0, 1, \text{no})$

return χ

4 Problem Instance Complexity

A *problem instance* in the case of computing character values of the symmetric group is simply an ordered pair (λ, μ) of partitions of the same integer.

4.1 *MurNak*

Let $\mathcal{R}_{\lambda, \mu}$ be the set of partitions appearing in the expansion of the right hand side of (1). More precisely, define

$$\mathcal{R}_{\lambda, \mu} = \left\{ \alpha \subseteq \lambda \mid \exists \alpha = \alpha_i \subset \alpha_{i-1} \subset \cdots \subset \alpha_0 = \lambda, \right. \\ \left. \xi_j = \alpha_{j-1} / \alpha_j \text{ is a rim hook, } |\xi_j| = \mu_j, 1 \leq j \leq i \right\}$$

which is the set of partitions one can obtain starting with λ by removing a sequence of rim hooks ξ_1, \dots, ξ_i of lengths μ_1, \dots, μ_i respectively, $i \leq \ell(\mu)$. In example 2, for instance,

$$\mathcal{R}_{(5,4,2,1), (4,3,2,2,1)} = \left\{ \begin{array}{c} \text{Diagram 1} \\ \text{Diagram 2} \\ \text{Diagram 3} \\ \text{Diagram 4} \\ \text{Diagram 5} \\ \text{Diagram 6} \\ \text{Diagram 7} \\ \text{Diagram 8} \\ \text{Diagram 9} \end{array}, \emptyset \right\}.$$

Denote $|\mathcal{R}_{\lambda, \mu}|$ by $r_{\lambda, \mu}$.

Proposition 8 *Let $t_{\text{MurNak}}(\lambda, \mu)$ be the running time of *MurNak* on input (λ, μ) . Then*

$$t_{\text{MurNak}}(\lambda, \mu) \in \Theta(r_{\lambda, \mu} h_{1,1}(\lambda)) \quad (8)$$

Where $h_{1,1}(\lambda) = \lambda_1 + \lambda'_1 - 1$ is the $(1, 1)$ hook number of λ .

PROOF. In computing $\text{MurNak}(\lambda, \mu)$, *MN1inner* is invoked precisely once for each node in the “recursion graph”. In each one of these $r_{\lambda, \mu}$ invocations of *MN1inner*, the length of its first parameter, R , is the same as the length of the essential part of the partition sequence of λ , which is $h_{1,1}(\lambda) + 1$.

Let $t_{\text{MN1inner}}(R, t)$ be the running time of *MN1inner*, excluding the recursion, on input $(R = R_1, \dots, R_s, t)$, $t \leq k$. *MN1inner* performs $\mu_t - 1$ iterations in the first loop and $s - \mu_t$ iterations in the second loop. In both loops, the time for each iteration is $\Theta(1)$. Therefore

$$t_{\text{MN1inner}}(R, t) \in \Theta(s).$$

It follows that each invocation of *MN1inner* during the computation of $\text{MurNak}(\lambda, \mu)$ takes time $\Theta(h_{1,1}(\lambda))$, with the possible exception of one invocation with the

trivial case $t > k$ which takes $\Theta(1)$. However, this possible exception is negligible as long as $r_{\lambda,\mu} > 1$.

Consequently,

$$\begin{aligned} t_{MurNak}(\lambda, \mu) &\in t_{PartSeq}(\lambda) + r_{\lambda,\mu} \Theta(h_{1,1}(\lambda)) \\ &= \Theta(h_{1,1}) + r_{\lambda,\mu} \Theta(h_{1,1}(\lambda)) \\ &= \Theta(r_{\lambda,\mu} h_{1,1}(\lambda)) \quad \square \end{aligned}$$

If $\lambda \vdash n$ then for any $\mu \vdash n$, $\mathcal{R}_{\lambda,\mu} \subseteq \mathcal{R}_{\lambda,(1^n)} = \{\alpha \mid \alpha \subseteq \lambda\}$ and consequently $r_{\lambda,\mu} \leq r_{\lambda,(1^n)}$. Define

$$r_\lambda = r_{\lambda,(1^n)},$$

the number of partitions α such that $\alpha \subseteq \lambda$.

Lemma 9 *Let $\lambda = (\lambda_1, \dots, \lambda_m) \vdash n$. Then*

$$r_\lambda = \det \left(\binom{\lambda_i + 1}{1 + i - j} \right)_{\substack{1 \leq i \leq m \\ 1 \leq j \leq m}}$$

The lemma follows from Stanley (1986), ch. 3, ex. 63: substituting the empty partition for μ and 1 for n , it states that

$$\zeta^2(\emptyset, \lambda) = \det \left(\binom{\lambda_i + 1}{1 + i - j} \right)_{\substack{1 \leq i \leq m \\ 1 \leq j \leq m}}.$$

By definition of ζ ,

$$\zeta^2(\mu, \lambda) = \sum_{\mu \leq \alpha \leq \lambda} 1.$$

Noting that the partial order \leq defined in the exercise coincides with containment of Young diagrams, we have

$$\zeta^2(\emptyset, \lambda) = \sum_{\emptyset \leq \alpha \leq \lambda} 1 = \sum_{\alpha \subseteq \lambda} 1 = \#\{\alpha \mid \alpha \subseteq \lambda\} = r_\lambda.$$

4.2 Roich

Let

$$\mathcal{Q}_{\lambda,\mu} = \{Q \mid Q \text{ is a standard tableau of shape } \alpha \subseteq \lambda \text{ and } g_\mu(Q, i) \neq 0, 1 \leq i \leq |\alpha|\}$$

which is the set of standard tableaux Q contained in λ such that the values $g_\mu(Q, 1), g_\mu(Q, 2), \dots, g_\mu(Q, j)$ alone, where $j = |\text{sh}(Q)|$, are insufficient to

determine whether $\prod_{i=1}^{n+1} g_\mu(T, i) = 0$ for all T such that $T|_j = Q$. (Hence, for each $Q \in \mathcal{Q}_{\lambda, \mu}$, *RoiInner* is invoked to compute $A(\lambda, \mu, Q)$).

For instance, in example 5,

$$\mathcal{Q}_{\lambda, \mu} = \left\{ \emptyset, \begin{smallmatrix} 1 \\ 1 \end{smallmatrix}, \begin{smallmatrix} 1 & 2 \\ 2 \end{smallmatrix}, \begin{smallmatrix} 1 \\ 2 \end{smallmatrix}, \begin{smallmatrix} 1 & 3 \\ 2 \end{smallmatrix}, \begin{smallmatrix} 1 \\ 2 \end{smallmatrix}, \begin{smallmatrix} 1 & 3 \\ 2 & 4 \end{smallmatrix}, \begin{smallmatrix} 1 & 4 \\ 2 & 3 \end{smallmatrix} \right\}.$$

Note that $S = \begin{smallmatrix} 1 & 2 \\ 3 \end{smallmatrix} \notin \mathcal{Q}_{\lambda, \mu}$, since $g_\mu(S, 3) = 0$. Consequently, the algorithm does not compute $A(\lambda, \mu, S)$.

Define

$$q_{\lambda, \mu} = |\mathcal{Q}_{\lambda, \mu}|.$$

$\mathcal{Q}_{\lambda, \mu}$ is just the set of non-leaf nodes in the recursion tree of *RoiInner*. Since the algorithm's work on each such node is linear in $\ell(\lambda)$, and the work on each leaf node is constant, we have

Proposition 10 *Let $t_{\text{Roich}}(\lambda, \mu)$ be the running time of Roich on input (λ, μ) . Then*

$$t_{\text{Roich}}(\lambda, \mu) \in \Theta(\ell(\lambda)q_{\lambda, \mu}). \quad (9)$$

It is clear from (3) that if $B(\mu) = \{1, 2, \dots, n\} = B((1^n))$ then $f_\mu(Q, i) \neq 0$ for every tableau Q and $1 \leq i < n$. Thus for any $\mu \vdash n$, $\mathcal{Q}_{\lambda, \mu} \subseteq \mathcal{Q}_{\lambda, (1^n)}$, and consequently $q_{\lambda, \mu} \leq q_{\lambda, (1^n)}$. Define

$$q_\lambda = q_{\lambda, (1^n)},$$

the number of standard tableaux of shapes contained in λ .

Lemma 11 *Let $\lambda \vdash n$. Then*

$$d_\lambda \leq q_\lambda \leq nd_\lambda + 1$$

where d_λ is the number of standard Young tableaux of shape λ .

PROOF. Let

$$\mathcal{D}_\lambda = \{Q \mid Q \text{ is a standard tableau of shape } \lambda\}.$$

Then

$$\mathcal{D}_\lambda \subseteq \mathcal{Q}_\lambda \subseteq \{\emptyset\} \cup \{T|_i \mid T \in \mathcal{D}_\lambda, 1 \leq i \leq n\}$$

and the lemma follows since $d_\lambda = |\mathcal{D}_\lambda|$. \square

5 Comparing the Algorithms

5.0 Worst case analysis

Recall that problem instances in the case of computing character values of the symmetric group, are simply pairs (λ, μ) of partitions of the same integer. In order to compare the algorithms' running times, we express them as functions of the *problem instance size*. A natural measure of instance size is the weight n of the partitions λ and μ .

In worst case analysis we consider the maximum running time of each of the algorithms on a problem instance of size n , namely

$$t_{MurNak}(n) = \max_{\lambda, \mu \vdash n} t_{MurNak}(\lambda, \mu)$$

and

$$t_{Roich}(n) = \max_{\lambda, \mu \vdash n} t_{Roich}(\lambda, \mu).$$

By proposition 8,

$$t_{MurNak}(n) \in \Theta(\max_{\lambda, \mu \vdash n} r_{\lambda, \mu} h_{1,1}(\lambda)) = \Theta(\max_{\lambda \vdash n} r_{\lambda} h_{1,1}(\lambda))$$

and by proposition 10,

$$t_{Roich}(n) \in \Theta(\max_{\lambda, \mu \vdash n} \ell(\lambda) q_{\lambda, \mu}) = \Theta(\max_{\lambda \vdash n} \ell(\lambda) q_{\lambda}).$$

Hence we seek expressions for (bounds on) $\max_{\lambda \vdash n} r_{\lambda} h_{1,1}(\lambda)$ and $\max_{\lambda \vdash n} \ell(\lambda) q_{\lambda}$ as functions of n .

We consider only certain families of problem instances, namely those in which λ is restricted to a given (k, ℓ) hook (see below).

5.1 (k, ℓ) hooks

The (k, ℓ) *hook* is the infinite shape $\{(i, j) \mid i \leq k \text{ or } j \leq \ell\}$. Let $H(k, \ell; n)$ be the set of all partitions of n lying inside the (k, ℓ) hook, that is

$$H(k, \ell; n) = \{\lambda \vdash n \mid \lambda_{k+1} \leq \ell\}.$$

The two propositions in this subsections show that for partitions in the (k, ℓ) hook, *MurNak* runs in polynomial time $\Theta(n^{k+\ell+1})$ whereas *Roich*'s running

time is exponential in n , being in $\Omega(n^{-g}(k+\ell)^n) \cap O(n^{-g+2}(k+\ell)^n)$ for some constant g .

Proposition 12 *Fix k and ℓ . Then*

$$\max_{\lambda \in H(k, \ell; n)} h_{1,1}(\lambda) r_\lambda \in \Theta(n^{k+\ell+1})$$

where $h_{1,1}(\lambda) = \lambda_1 + \lambda'_1 - 1$.

The proof requires the following lemmas. We use the notations

$$\begin{aligned} a \vee b &= \max\{a, b\} \\ a \wedge b &= \min\{a, b\}. \end{aligned}$$

Lemma 13 *Fix k and ℓ . If $\lambda \in H(k, \ell; n)$ then $h_{1,1}(\lambda) \in \Theta(n)$.*

PROOF.

$$\lambda \subset \{(i, j) \mid 1 \leq i \leq k, 1 \leq j \leq \lambda_1\} \cup \{(i, j) \mid 1 \leq j \leq \ell, 1 \leq i \leq \lambda'_1\}$$

so

$$n = |\lambda| \leq k\lambda_1 + \ell\lambda'_1 \leq (k \vee \ell)(\lambda_1 + \lambda'_1) \leq 2(k \vee \ell)h_{1,1}(\lambda).$$

On the other hand,

$$H_{1,1}(\lambda) \subseteq \lambda$$

so

$$h_{1,1}(\lambda) = |H_{1,1}(\lambda)| \leq n.$$

Therefore

$$\frac{1}{2(k \vee \ell)} n \leq h_{1,1}(\lambda) \leq n. \quad \square$$

Lemma 14 (The k strip ($\ell = 0$) case) *Let $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_k) \vdash n$. Then*

$$\frac{1}{k!} \prod_{i=1}^k (\lambda_i + 1) \leq r_\lambda \leq \prod_{i=1}^k (\lambda_i + 1). \quad (10)$$

PROOF. Set

$$T_\lambda = \{(p_1, \dots, p_k) \in \mathbb{Z}^k \mid 0 \leq p_i \leq \lambda_i\}.$$

Let $p = (p_1, \dots, p_k) \in T_\lambda$. Then there exists a permutation $\sigma \in S_k$ such that $p_{\sigma(1)} \geq p_{\sigma(2)} \geq \dots \geq p_{\sigma(k)}$. We claim that $\alpha = \sigma p = (p_{\sigma(1)}, p_{\sigma(2)}, \dots, p_{\sigma(k)}) \in \mathcal{R}_\lambda$. Indeed, α is a partition, and for all i we have

$$|\{j \mid \alpha_j > \lambda_i\}| = |\{j \mid p_{\sigma(j)} > \lambda_i\}| = |\{j \mid p_j > \lambda_i\}| < i$$

whence $\alpha_i \leq \lambda_i$.

It follows that for every $p \in T_\lambda$ there exist $\alpha \in \mathcal{R}_\lambda$ and $\sigma \in S_k$ such that $p = \sigma^{-1}\alpha$. Thus $T_\lambda \subseteq S_k \mathcal{R}_\lambda$, so

$$\begin{aligned} |T_\lambda| &\leq |S_k| \cdot |\mathcal{R}_\lambda| \\ \prod_{i=1}^k (\lambda_i + 1) &\leq k! r_\lambda \\ \frac{1}{k!} \prod_{i=1}^k (\lambda_i + 1) &\leq r_\lambda. \end{aligned}$$

The other inequality in (10) follows from the fact that $\mathcal{R}_\lambda \subseteq T_\lambda$. \square

PROOF. [Proof of proposition 12] Without loss of generality, assume $k \leq \ell$. We have

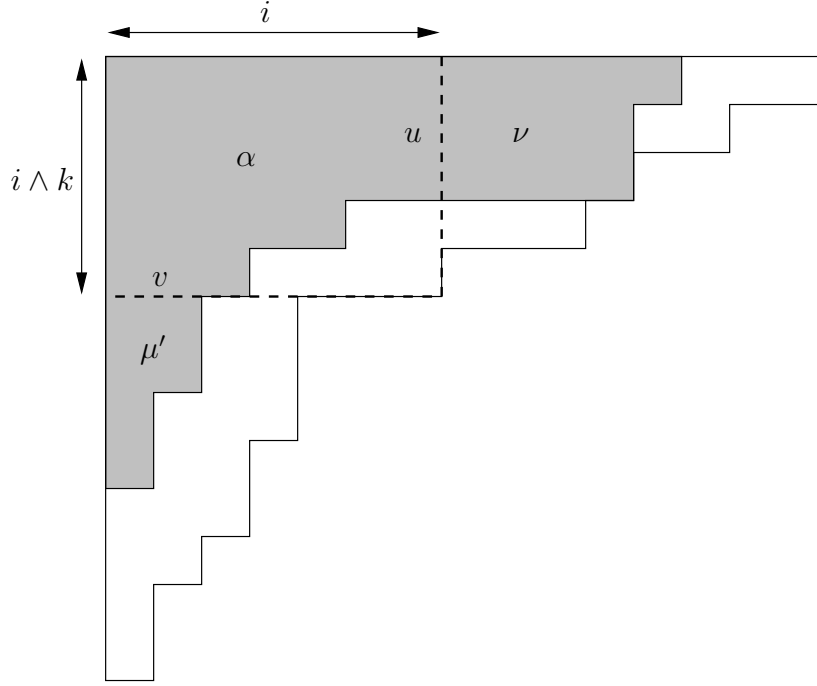
$$H(k, \ell; n) = \bigcup_{r=1}^{\ell} H_i \tag{11}$$

where

$$H_i = \{\lambda \in H(k, \ell; n) \mid (i \wedge k, i) \in \lambda, ((i+1) \wedge k, i+1) \notin \lambda\}.$$

Let $\lambda \in H_i$. There is a bijection between \mathcal{R}_λ and ordered triplets of partitions (α, ν, μ) such that

1. α is contained in the $(i \wedge k) \times i$ rectangle, that is $\alpha \in \mathcal{R}_{(i \wedge k)}$.
 2. $\nu \subseteq (\lambda_1 - i, \lambda_2 - i, \dots, \lambda_u - i)$ where $u = \max\{j \mid \alpha_j = i\}$.
 3. $\mu \subseteq (\lambda'_1 - (i \wedge k), \lambda'_2 - (i \wedge k), \dots, \lambda'_v - (i \wedge k))$ where $v = \max\{j \mid \alpha'_j = i \wedge k\}$.
- The following figure illustrates this bijection, showing $\lambda \in H_i$ in white and $\rho \in \mathcal{R}_\lambda$ shaded:



It follows that

$$r_\lambda = |\mathcal{R}_\lambda| = \sum_{\substack{0 \leq u \leq i \wedge k \\ 0 \leq v \leq i}} a_{u,v} r_{(\lambda_1-i, \lambda_2-i, \dots, \lambda_u-i)} r_{(\lambda'_1-(i \wedge k), \lambda'_2-(i \wedge k), \dots, \lambda'_v-(i \wedge k))}$$

where

$$a_{u,v} = \left| \left\{ \alpha \in \mathcal{R}_{(i \wedge k)} \mid \max\{j \mid \alpha_j = i\} = u, \max\{j \mid \alpha'_j = i \wedge k\} = v \right\} \right|.$$

If $\alpha \in \mathcal{R}_{(i \wedge k)}$ and u, v are as above, then $u = i \wedge k \iff v = i \iff \alpha = (i^{i \wedge k})$, thus $a_{i \wedge k, i} = 1$. Otherwise, the partition sequence of α is $\dots \underbrace{\mathbf{0} \mathbf{1} \dots \mathbf{1}}_{v \text{ 1s}} \mathbf{0} \tilde{A} \mathbf{1} \underbrace{\mathbf{0} \dots \mathbf{0}}_{u \text{ 0s}} \mathbf{1} \dots$

where \tilde{A} is any sequence containing exactly $(i - v - 1)$ **1**s and $((i \wedge k) - u - 1)$ **0**s, whence $a_{u,v} = \binom{(i \wedge k) - u - 1 + i - v - 1}{i - v - 1}$ for $u < i \wedge k, v < i$.

Thus

$$\begin{aligned} r_\lambda = |\mathcal{R}_\lambda| &= \sum_{\substack{0 \leq u < i \wedge k \\ 0 \leq v < i}} \binom{i \wedge k + i - u - v - 2}{i - v - 1} r_{(\lambda_1-i, \lambda_2-i, \dots, \lambda_u-i)} r_{(\lambda'_1-(i \wedge k), \lambda'_2-(i \wedge k), \dots, \lambda'_v-(i \wedge k))} \\ &\quad + r_{(\lambda_1-i, \lambda_2-i, \dots, \lambda_{i \wedge k}-i)} r_{(\lambda'_1-(i \wedge k), \lambda'_2-(i \wedge k), \dots, \lambda'_{i \wedge k}-(i \wedge k))} \end{aligned}$$

Since $\binom{i \wedge k + i - u - v - 2}{i - v - 1} > 0$ for all values of u and v in the sum and does not

depend on λ ,

$$\begin{aligned}
r_\lambda &\in \Theta \left(\sum_{\substack{1 \leq u \leq i \wedge k \\ 1 \leq v \leq i}} r_{(\lambda_1-i, \lambda_2-i, \dots, \lambda_u-i)} r_{(\lambda'_1-(i \wedge k), \lambda'_2-(i \wedge k), \dots, \lambda'_v-(i \wedge k))} \right. \\
&\quad \left. + r_{(\lambda_1-i, \lambda_2-i, \dots, \lambda_{i \wedge k}-i)} r_{(\lambda'_1-(i \wedge k), \lambda'_2-(i \wedge k), \dots, \lambda'_i-(i \wedge k))} \right) \\
&\text{(by lemma 14)} \\
&= \Theta \left(\sum_{\substack{1 \leq u \leq i \wedge k \\ 1 \leq v \leq i}} \prod_{s=1}^u (\lambda_s - i + 1) \prod_{t=1}^v (\lambda'_t - (i \wedge k) + 1) \right. \\
&\quad \left. + \prod_{s=1}^{i \wedge k} (\lambda_s - i + 1) \prod_{t=1}^i (\lambda'_t - (i \wedge k) + 1) \right) \\
&= \Theta \left(\prod_{s=1}^{i \wedge k} (\lambda_s - i + 1) \prod_{t=1}^i (\lambda'_t - (i \wedge k) + 1) \right)
\end{aligned}$$

Whence

$$\max_{\lambda \in H_i} r_\lambda \in \Theta \left(\left(\frac{n - i(i \wedge k)}{i + (i \wedge k)} \right)^{i+(i \wedge k)} \right) = \Theta(n^{i+(i \wedge k)})$$

and therefore, by (11),

$$\max_{\lambda \in H(k, \ell; n)} r_\lambda = \max_{1 \leq i \leq \ell} \max_{\lambda \in H_i} r_\lambda \in \Theta(n^{k+\ell})$$

Finally, by lemma 13

$$\max_{\lambda \in H(k, \ell; n)} h_{1,1}(\lambda) r_\lambda \in \Theta(n^{k+\ell+1}) \quad \square$$

The running time of *Roich* for partitions in the (k, ℓ) hook is determined up to a factor of order n^2 in the following proposition.

Proposition 15 *Fix k and ℓ . Then*

$$\max_{\lambda \in H(k, \ell; n)} \ell(\lambda) q_\lambda \in \Omega \left(\left(\frac{1}{n} \right)^g (k + \ell)^n \right) \cap O \left(\left(\frac{1}{n} \right)^{g-2} (k + \ell)^n \right)$$

for a certain constant g .

The proposition follows immediately from lemma 11 and from the following theorem.

Theorem 16 (Regev (1998), Theorem 3.3 (4)) *Assume n is large and $\lambda \in H(k, \ell; n)$ maximizes d_λ . There exist constants c and g such that*

$$d_\lambda \simeq c \left(\frac{1}{n} \right)^g (k + \ell)^n.$$

Example 17 *Table 1 shows the running times of the two algorithms on $(\lambda, (1^{|\lambda|}))$ for several λ s in the $(1, 2)$ hook. Maximal $r_{\lambda}h_{1,1}(\lambda)$ and $\ell(\lambda)q_{\lambda}$ values for each n appear in boldface.*

5.2 General diagrams

Table 2 shows the running times of the two algorithms for several pairs (λ, μ) . The values of $r_{\lambda, \mu}$ and $q_{\lambda, \mu}$ were obtained by running *MurNak* and *Roich* on each pair and counting invocations of *MN1Inner* and *RoiInner* respectively. Maximal $r_{\lambda, \mu}h_{1,1}(\lambda)$ and $\ell(\lambda)q_{\lambda, \mu}$ values for each n appear in boldface.

Acknowledgments

This paper is based on work conducted for my M.Sc. thesis, under the supervision of Professor Amitai Regev. I would like to thank him for his patient guidance, helpful advice and constant encouragement, and specifically for reviewing and commenting on drafts of this paper. I would also like to thank Yuval Roichman for his comments.

References

- Barcelo, H., Ram, A., 1999. Combinatorial Representation Theory, New perspectives in algebraic combinatorics (Berkely, CA, 1996–1997), 23–90, Math. Sci. Res. Inst. Publ. 38, Cambridge University Press, Cambridge.
- Bessenrodt, C., 1998. On hooks of Young diagrams, *Ann. Combin.* 2, 103–110.
- Kazhdan, D., Lusztig, G., 1979. Representations of Coxeter groups and Hecke algebras, *Invent. Math.* 53, 165–184.
- Murnaghan, F. D., 1937. The characters of the symmetric group, *Amer. J. Math.* 59, 739–753.
- Nakayama, T., 1940. On some modular properties of irreducible representations of a symmetric group I and II, *Jap. J. Math.* 17, 165–184, 411–423.
- Olsson, J. B., 1993. Combinatorics and representations of finite groups, *Vorlesungen aus dem FB Mathematik der Univ. Essen*, Heft 20.
- Regev, A., 1998. Maximal degrees for Young diagrams in the (k, l) hook, *European J. Combin.* 19, 721–726.
- Roichman, Y., 1997. A recursive rule for Kazhdan-Lusztig characters, *Adv. Math.* 129, No. 1, 25–29.
- Sagan, B., 1991. *The Symmetric Group*, Wadsworth and Brooks, Pacific Grove, California.
- Stanley, R. P., 1986. *Enumerative Combinatorics*, vol. 1, Wadsworth and Brooks, Monterey, California.

Table 1

Running times of the two algorithms on $(\lambda, 1^{|\lambda|})$ when λ s in the (1,2) hook.



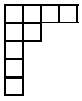
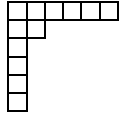
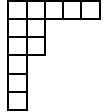
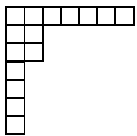
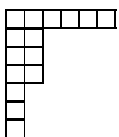
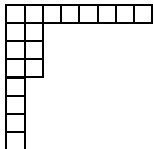
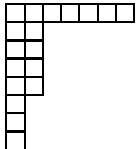
n	λ	Murnaghan-Nakayama		Roichman	
		r_λ	$r_\lambda h_{1,1}(\lambda)$	q_λ	$\ell(\lambda)q_\lambda$
6		13	78	35	105
		14	70	48	144
9		33	264	599	2,995
12		62	682	7,010	42,060
		67	670	11,664	69,984
15		116	1,508	170,566	1,193,962
		118	1,416	238,174	1,667,218
18		191	2,865	4,000,428	32,003,424
		189	2,646	5,029,991	40,215,928

Table 2

Running times of the two algorithms for various inputs.

n	λ	μ	Murnaghan-Nakayama		Roichman	
			$r_{\lambda,\mu}$	$r_{\lambda,\mu}h_{1,1}(\lambda)$	$q_{\lambda,\mu}$	$\ell(\lambda)q_{\lambda,\mu}$
6		(1^6)	14	70	48	144
			5	25	32	96
		(1^6)	13	78	35	140
8		(1^8)	26	182	276	1104
			7	49	97	485
12		(1^{12})	75	675	22,454	112,270
			1	9	1,912	9,560
		(1^{12})	62	682	7,010	42,060
		(1^{12})	63	504	13,921	69,605
		(1^{12})	9	72	1,384	6,920
		(1^{12})	9	72	1,384	6,920
15		(1^{15})	139	1,390	714,201	4,285,206
		(1^{15})	142	1,704	463,996	3,247,972